To appear in *Advanced Robotics*
Vol. 00, No. 00, April 2017, 1–12

FULL PAPER

Morphological computation based control of a modular, pneumatically driven, soft robotic arm

M. Eder[a], F. Hisch[b] and H. Hauser[c,d]*

[a] *Artificial Intelligence Laboratory, Institute for Informatics, University of Zurich, Switzerland*; [b] *Institut für Informatik, Technische Universität München, Germany*; [c] *Department of Engineering Mathematics, University of Bristol, UK*; [d] *Bristol Robotics Laboratory,UK*

(*v1.0 released January 2013*)

The dynamics of soft robotic bodies are typically complex and exhibit nonlinearities and a high-dimensional state space. As a result, such systems are difficult to model and, therefore, hard to control. In this work we use a model-free approach by employing the concept of morphological computation, which understands the complexity of the dynamics of such bodies as potential computational resources that can be exploited, for example, for control. The validity of this approach has been previously demonstrated in a number of simulations as well on a number of simple soft robotic platforms. However, this work takes the approach a significant step further by implementing it on a highly complex pneumatically driven robotic arm consisting of multiple modular segments, bringing the morphological computation based control approach closer to real industrial applications. We demonstrate that various end point trajectories can be learned and be reproduced consistently in a remarkably robust fashion. The presented morphological computation setup needs no model of the highly complex robot. Moreover, by exploiting the seemingly unbeneficial complex dynamics as a computational resource, the learning task to implement a nonlinear and dynamic control can be reduced to simple linear regression.

Keywords: morphological computation, soft robotics, compliant robot arm, model-free control, embodiment

1. Introduction

In recent years there has been an increasing interest in building soft-bodied robots [1]. There are a number of good reasons to use soft materials in robot design, including their great potential to be used to build robust, resilient, adaptive, and energy efficient systems. Another beneficial aspect is the inherent safety aspect [2]. Possible applications of such systems include close human-robot collaboration scenarios, environments with constrained spaces, or service robots – all particularly interesting for future industrial environments. However, soft-bodied designs have not yet found their way into factories. One of the main reasons is their unsolved control problem. Soft structures typically exhibit complex, nonlinear dynamics, which make it challenging to model such systems and, therefore, very difficult to control. Current, more traditional systems, i.e., robots using rigid body parts and high-torque servomotors, don't face this control problem. Their rigid structures are designed to facilitate the modeling process and, consequently, their control. However, they are also potentially more dangerous in collaborative interaction scenarios. As a consequence, such robots are typically separated from humans by fences and a direct collaboration is prevented. There exist a number of approaches to circumvent this limitation

*Corresponding author. Email: helmut.hauser@bristol.ac.uk

by negotiating a trade-off between controllability and safety for conventional, rigid robots. This includes speed restrictions [3], installation of light curtains [3], monitoring of operator [4] and others [3–6]. However, these setups are rather complex resulting in additional costs and further constraints on the collaborations between robots and humans [7]. If we are able to solve the control problem of soft structures, soft robotics has the great potential to serve as the framework to design close human-robot interaction systems.

One possible solution is to take a closer look at biological systems, which, as soft robots, are also inherently compliant and exhibit rich and complex dynamics. It seems the evolutionary process was able to produce solutions that are able to cope with such complexity in a very impressive manner. This is especially true considering that animals outperform their robotic counter parts in almost any task. One of the main differences between robots and biological systems is the way they are controlled. In robotics systems we have a body that is fully dominated by a central controller. In biological systems we can observe a more decentralized approach, offloading big parts of the computational costs to morphological properties like shape, form and dynamical parameters. This is often referred to as morphological computation [8]. While the concept is quite abstract and there is no commonly accepted definition, nevertheless, there exist rigorous approaches using mathematical frameworks to describe the computational power of morphology [9–11].

Interestingly, in the context Hauser et al. [9, 10], the aforementioned complex dynamics, which have been perceived so far as a disadvantage of soft-bodied machines, are considered beneficial in this framework. They can be exploited as a computational resource, which can result in simpler controllers. One could say, loosely speaking, a morphological computation based control approach outsources necessary control computations to the physical body of the robot [12]. Hence, the complexity of the controller is offloaded to the robot's body, leaving the controller on top with a much simpler task.[1]

While Hauser et al. [9, 10] demonstrated the concept in simulations, the applicability on real robotics platforms has been proven as well. For example, Nakajima et al. used in a series of publications [14–16] a soft silicone arm and its dynamics as a computational resource. Caluwaerts et al. [17] used the setup to develop controllers for tensegrity structures in the context of explanatory exploration. Another example of using the physical body as a computational resource for control was carried out by Zhao et al. [18]. They exploited a soft bio-inspired spine structure of a quadruped robot to control its own locomotion. While these examples are interesting proofs of concept, they are far from the complexity required in a more industrial application.

The work presented here takes the approach one step further by employing the theoretical framework on a complex, modular, soft robotic systems, which is meant to be used in an industrial environment. We show the successful exploitation of this complex arm and its dynamics as a computational resource to control it. With the help of its morphology the setup will learn to move along various end effector trajectories, which are designed to mimic real industrial application within the context of human-robot interaction. We explore limitations of the approach and discuss their implications and potential solutions.

In a next section, we provide the reader with more details on theoretical background of the morphological computation setup in general. The implementation section describes the used platform, technical details, and how the control framework has been implemented. In the results section we present various experiments demonstrating the feasibility of the approach and identifying its limitations. Finally, we discuss implications and future work.

[1] It the most extreme case nonlinear control can be even outsourced entirely to the body, like in the case of Passive Dynamic Walkers [13].

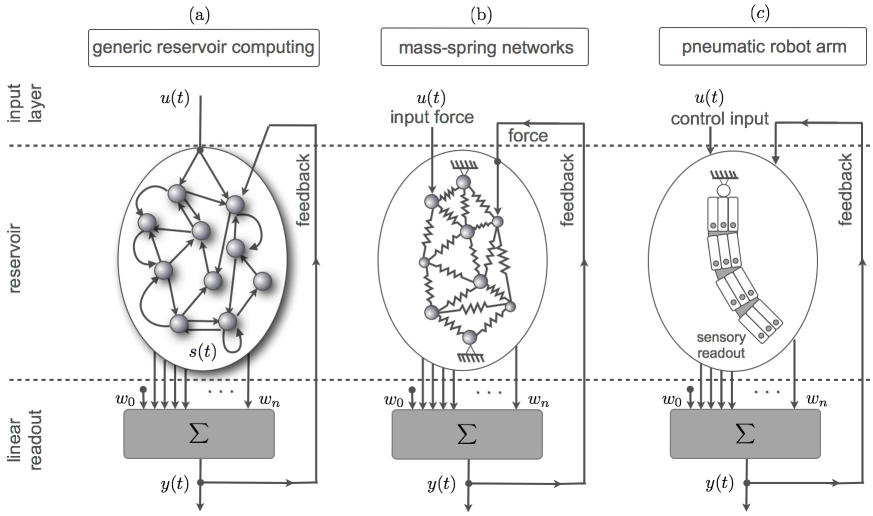


Figure 1. Comparison of various implementations of reservoir computing (a) Generic setup of reservoir computing (b) Implementation with a network of nonlinear mass-spring-damper systems as described in [9, 10] **(c) Employing the complex dynamics of a pneumatically driven, modular robot arm as a reservoir as proposed in this work.**

2. Theoretical background on Morphological Computation

The underlying principle of the approach is a machine learning technique called *reservoir computing*, which has been highly successful in emulating complex dynamical systems [19]. At its core there is a complex, high-dimensional nonlinear dynamical system, called the *reservoir*, see Figure 1a, **which is exploited as computational resource**. **Interestingly,** the reservoir can be implemented in many different ways, as long as it's sufficiently dynamic, nonlinear and exhibits a high-dimensional state space [12]. Hauser et al. used this insight to demonstrate that one can build a reservoir with a network of nonlinear mass-spring-damper systems (see Figure 1b). **However, as we show in this work the complex dynamics of a pneumatically driven robot arm can be exploited as a computational resource, i.e., as a reservoir, as well (see Figure 1c).**

In all three cases, when a typically low-dimensional input $u(t)$ is applied, the reservoir responds with a high-dimensional set of internal signals, i.e., the state of the dynamical system $\mathbf{s}(t)$. These signals will reflect the current input, but they will also represent the input history as well nonlinear combinations of these signals. The reservoir will act as a *kernel* (in the machine learning sense), i.e., it will project nonlinearly the low-dimensional input in its high-dimensional state space. These signals (or a subset of them), here denoted as $\mathbf{s}(t)$, can be then read out **(e.g., with sensors)** and linearly combined to produced a desired output $y(t)$. The remarkable property of this approach is that it can be used to learn to emulate complex input-output relationships (e.g., nonlinear dynamical systems**, e.g., a controller**) by only finding optimal output weights $\mathbf{w}^*$, without changing the properties of the reservoir. This means, with the help of the reservoir the learning task to emulate complex, nonlinear dynamic systems can be reduced to simple linear regression.

As Figure 1 shows the output $y(t)$ can also be fed back into the system. It has been shown in [10] that this feedback loop drastically improves the computational power by allowing the implementation of non-fading behavior, like robust nonlinear limit cycles, bifurcation, etc.

In this particular work we employ the concept of reservoir computing to a pneumatically driven robot arm (see Figure 1c). The morphological structure of the arm and its complex nonlinear dynamics are exploited as a computational resource

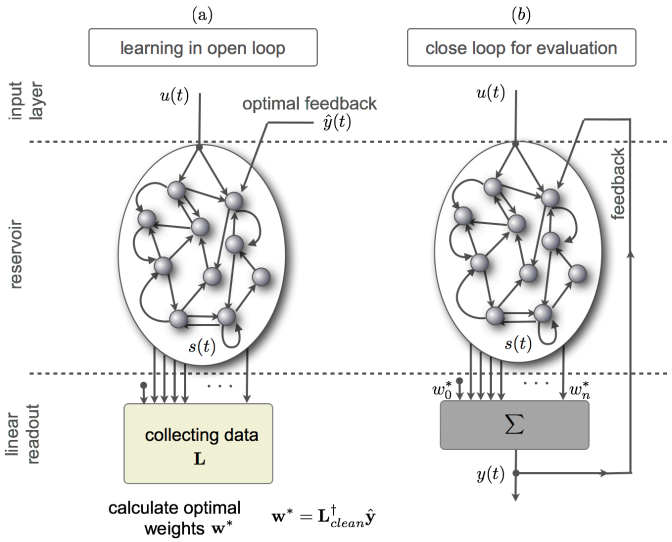


Figure 2. **General learning setup in reservoir computing with feedback loop. The setup is supervised, i.e., input time series $u(t)$ and corresponding optimal target output $\hat{y}(t)$ are given (a) In open loop the reservoir is excited by the time varying input $u(t)$ and the optimal feedback signal, i.e., the target readout $\hat{y}(t)$. The readout from the state $\mathbf{s}(t)$ is collected in a matrix $\mathbf{L}$ to calculate offline the optimal readout weights $\mathbf{w}^*$ (b) Using these constants weights the loop can be closed and the desired computation is carried out.**

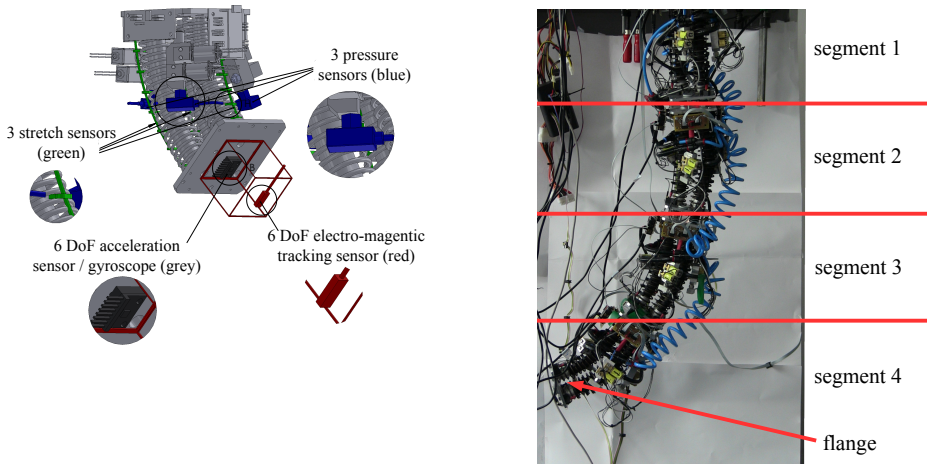
(i.e., the reservoir) to learn to emulate a nonlinear dynamic controller. The input/feedback is provided to the system in form of control signals for the valves. The readout is established through a total of 48 **sensors**[1] **and the output is 2D trajectory of the end-effector project to the ground. For more details on the robot arm setup we refer to Section 3.**

The learning process is based on a supervised learning setup, i.e., it is assumed that both the input time series $u(t)$ and the corresponding desired output $\hat{y}(t)$, i.e., the target, are given. The learning task is to find an optimal set of output weights $\mathbf{w}^*$ that produces the target output as close as possible by linearly combing the readout signals $\mathbf{s}(t)$ from the reservoir, i.e., linearly combining the sensor signals.
Since we have a feedback loop the learning is carried out in open loop (see Figure 2). The reservoir is excited by the given input time series $u(t)$ **and the optimal feedback, which is the target output** $\hat{y}(t)$. The reservoir, being a dynamical system, responds with the change of its n-dimensional state. The time series of the states (or a subset) are read out via sensors over k time steps as a set of signals $s_i(t) \quad \forall \quad i = 0, 1, \ldots, n$ (with s_0 being a constant value to provide an offset) and are collected in a $n \times k$ matrix $\mathbf{L}$. To get rid of any initial transients a certain number of l time steps from the beginning are thrown out (the so-called washout), reducing the data matrix $\mathbf{L}$ to a $n \times (k - l)$ matrix $\mathbf{L}_{clean}$.
In addition, the target output is collected within a vector $\hat{\mathbf{y}}$ over the same amount of k time steps and reduced by **the same** l time steps. The optimal output weights $\mathbf{w}^* = [w_0, w_1, \ldots, w_n]^T$, with w_0 representing a constant bias, can be calculated by

$$\mathbf{w}^* = \mathbf{L}^{\dagger}_{clean}\hat{\mathbf{y}}, \tag{1}$$

with $\mathbf{L}^{\dagger}_{clean}$ being the Moore-Penrose pseudoinverse as $\mathbf{L}_{clean}$ in general is not a square matrix. The extension to multiple inputs and outputs is straight forward by introduction a $k \times i$ matrix $\mathbf{U}$ for the input and a $k \times j$ matrix $\hat{\mathbf{Y}}$ for the output, with i and j being the number of inputs and outputs respectively.
After learning the optimal weights the loop can be closed and the system evaluated.

[1]Note that the actual state of the system is higher, but often a subset of sensory information can be sufficient.



(a) (b)

Figure 3. The used robot arm (a) One of the four arm segments including actuators and sensors. Three pneumatic artificial muscles move the segment. The measurements of the states are realized by 3 pressure sensors, 3 stretch sensors, one 6-DOF acceleration sensors, and 1 electromagnetic tracking sensor (only on distal segment) (b) The entire soft robotic arm with 4 segments in a deflected state

Note that the weights are fixed after the learning process and they provide only a linear and static mapping of the internal state to the output.
The setup can be slightly adapted to use it to emulate a controller that controls the body itself. The resulting setup has no input $u(t)$ and only the feedback loop is used to control the system to keep the trajectory defined by the desired output $\hat{y}(t)$. This has been previously demonstrated with an octopus inspired silicon arm [15] as well as on a quadruped robot [18]. In this case the same learning setup can be employed. The paper uses exactly this setup (i.e., only feedback and no input) with a pneumatic robot arm. More details on the implementation are presented in the next section.

3. Experimental Setup and Implementation

This section describes the implementation of the morphological computation based approach to a specific soft robotic arm, see Figure 3. First, technical details on the robotic arm are given. Second, details on the learning and control implementation are provided.

3.1 *The Soft Robotic Arm*

All experiments were performed on a modular robotic arm developed by Eder et al. [20–22], see Figure 3. **For the sake of brevity we describe the physical platform only shortly here. For a complete description of the arm we refer to [22]**. The robot consists of four independent segments, which form a **continuum-style worm-like chain. The overall length of the entire robot is** $1050mm$**, its weight about** $4.8kg$ **and the maximum payload is approximately** $1.2kg$. Each of the segments of the robot (see Figure 3a for a detailed view) consists of three pneumatic artificial muscles (PAM, see [23] for more details), mounted between a base and an end-effector plate. Accordingly, each segment has three pressure

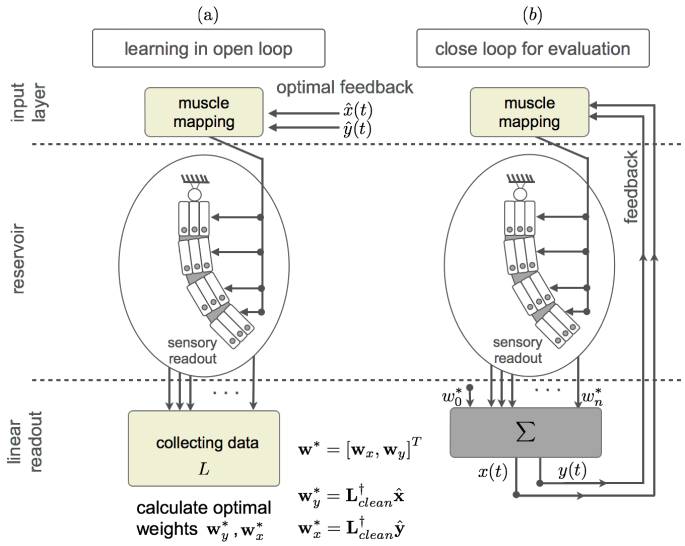


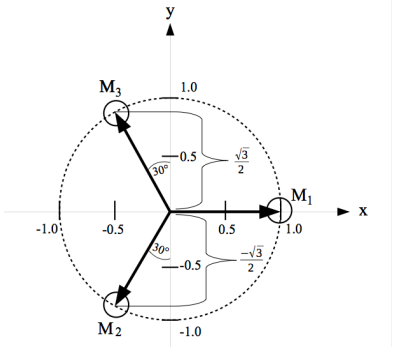
Figure 4. **Learning setup with the pneumatically driven robot arm.**

controller boards and the corresponding valves and one mainboard processor. Furthermore, a segment features three springs to stabilize the artificial muscles. Regarding sensing, i.e., to read out the (partial) state of the system, each segment has 3 stretch sensors, which measure the length of the individual PAMs, 3 pressure sensors, which measure the internal PAM pressures, and one 6D-Acceleration sensor (acceleration and gyroscope; 3 degree of freedom each) mounted at the segment's end-effector-plate. Each segment has 3 DOFs (degrees of freedom): two DOFs for bending, i.e. pan and tilt and one DOF of contraction/extension. By controlling the contraction of the muscles the segment bends. However, this also influences its adjacent segments, which makes **it non-trivial** to control. **Hence, conventional control approaches are difficult to apply**. Consequently, the full robot arm, as shown in Figure 3b, has $n = 48$ sensors (counting each sensor dimension of the 6D-acceleration sensor separately) and $m = 12$ actuators (PAMs). Additionally, an electromagnetic tracking sensor was mounted on the distal segment to identify the absolute position of the end point of the robot arm to evaluate the control result.

3.2 *Reservoir Computing Implementation on the Robot Arm*

In order to demonstrate the applicability of the **reservoir computing** approach as well as to investigate **its** robustness and limitations, we used the setup to learn to follow a range of two-dimensional target trajectories that are given in Cartesian coordinates $\hat{x}(t)$ an $\hat{y}(t)$. **The robot arm is exploited as a reservoir to produce continuously these two trajectories by linearly combing the current sensor values.** These means **the setup** needs to produce two outputs, $x(t)$ and $y(t)$ **(compare Figure 4)**. Accordingly, we have to find two sets of optimal output weights $\mathbf{w}_x^*$ and $\mathbf{w}_y^*$. **Each one of them are used independently to** linearly combine all available sensor values (a total of 48) to produce the two output target signals $\hat{x}(t)$ an $\hat{y}(t)$ as closely as possible. **These signals are** then fed back to the input layer(compare Figure 4).

At the input the coordinates have to be converted into control signals for the pressure for the pneumatic muscles , which are 3 per segment, i.e., M_1, M_2 and M_3, **see muscle mapping in Figure 4**. The mapping **is based on a highly simplified kinematic model and does not consider any dynamic properties of the system at all**. The corresponding equations can be seen in Figure 5. Note that these values are pressures and, hence, can take on only positive values. Furthermore, the values are rescaled to fit within the range of possible pressures. All four segments of the arm receive exactly the same values for their 3 muscles. Consequently, working as a kinematic chain they will bend all into the same direction.
As previously described, the learning requires a data collection process in open loop. In our



$$
\begin{aligned}
M_1 &= x + \frac{y}{\sqrt{3}} + M_2 \\
M_2 &= \max\left(-1 \ \min\left(x + \frac{y}{\sqrt{3}}, \frac{2y}{\sqrt{3}}\right), \ 0\right) \\
M_3 &= \frac{2y}{\sqrt{3}} + M_2
\end{aligned}
$$

Figure 5. Translation of Cartesian coordinates x and y into pneumatic muscle actuation, i.e., muscle pressures $M1$, $M2$ and $M3$. On the left the spacial arrangement of muscle actuators per segment (top view) is depicted. All 3 PAM actuators are equally distributed around the center. **The same three muscle pressures are applied to all segments.**

setup there is no input given, but instead the system is driven by the target feedback, i.e., $\hat{x}(t)$ and $\hat{y}(t)$ **(see Figure 4)**. These trajectories are converted directly into muscle pressures $M_1(t)$, $M_2(t)$ and $M_3(t)$ for all segments, which move the arm. The readout of all 48 sensors $s_i(t)$ with $i = 0, 1, \ldots, 48$, with s_0 being a constant to represent a bias, are collected over $k = 750$ time steps (each 0.1s)[1] in the $(48 + 1) \times 750$ matrix $\mathbf{L}$. A good washout time was found to be 250 time steps leaving us with reduced data matrix $\mathbf{L}_{clean}$. The optimal weights for both outputs $\mathbf{w}_x$ and $\mathbf{w}_y$ were found as previously described by inverting the reduced data matrix $\mathbf{L}_{clean}$ and multiplying it with the target outputs, i.e., $\mathbf{w}_x^* = \mathbf{L}_{clean}^{\dagger}\hat{\mathbf{x}}$ and $\mathbf{w}_y^* = \mathbf{L}_{clean}^{\dagger}\hat{\mathbf{y}}$.

case	x(t)	y(t)	trajectory
circular	$-\sin\left(t\frac{2\pi}{75}\right)$	$\cos\left(t\frac{2\pi}{75}\right)$	
axial	$-\sin\left(t\frac{2\pi}{75}\right)$	0	
oval	$-\sin\left(t\frac{2\pi}{75}\right)$	$d\cos\left(t\frac{2\pi}{75}\right)$	
spiral	$e^{-t\frac{2\pi}{1500}}\sin\left(t\frac{2\pi}{75}\right)$ +0.5	$e^{-t\frac{2\pi}{1500}}\cos\left(t\frac{2\pi}{75}\right)$ +0.5	

Table 1. 2D target trajectories used for the evaluation of the approach.

4. Results

This section presents the results of the experiments for learning various trajectories of the robot arm, compare Table 1, and testing the robustness of the learned control. To be more specific, the implemented trajectories were circle motions, circle motions with fixed end-effector plate, circle motions with additional weights (during learning, during evaluation, and both learning and evaluation), axial motions, oval motions, and spiral motions.[1] Each result is represented by three subfigures, see, e.g., Figure 6a. **For all figures in the results section, i.e., Figure 6, 7 and 8, the top rows show the produced outputs over time, i.e., the Cartesian coordinates $x(t)$ and $y(t)$ and the bottom rows show the same outputs, but in 2D space with produced trajectories in read, full lines and the target trajectory in black, dash-dotted lines. The middle rows show the measured muscle pressures M_1, M_2 and M_3.**

The experiments with a circular target trajectory (as defined in Table 1) are summarized in Figures 6 and 7. The success of the learning phase of our circular motion can be seen in Figure 6a. The corresponding 2D plot shows the produced trajectories based on the optimal found weights using the data matrix $\mathbf{L}_{clean}$ acquired during the learning phase (i.e., the feeback loop is not closed, 'offline performance'). The actual evaluation in closed loop (on-line performance) using the optimal weights $\mathbf{w}_x^*$ and $\mathbf{w}_y^*$ can be seen in Figure 6b. To test the robustness of the approach we fixed the end point of the robot to its resting position (i.e., in the middle of the 2D plot) during the first 250 time steps of the evaluation phase to release and observe the response of the system, see Figure 6c. As one can see, the robot does not actively control against the error, which is actually beneficial in the context of safe human-robot interaction. A traditional PID controllers would steadily increase the control effort in its attempt to reduce the error resulting in high and dangerous forces. Once the disruption ends at $t = 250$, the robot returns to its desired trajectory. However, to do so, it is necessary to manually move the robot into its learned trajectory, e.g. by giving the robot a short impulse. Again this could been seen as an (emergent) feature of the controller being a convenient way to tell the robot, now it's OK to move back on to your nominal trajectory.

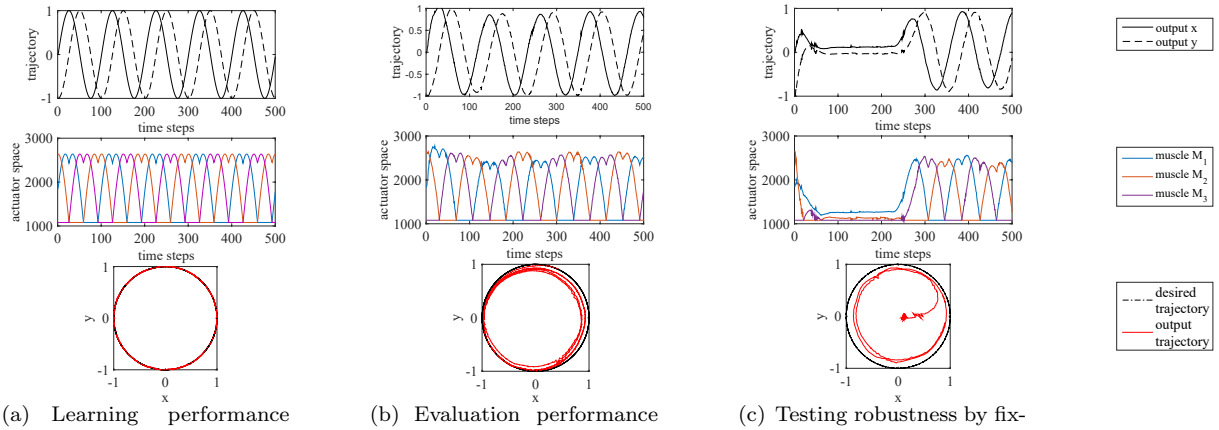
Another robustness test has been carried by adding weight of 250 g to the final segment of the robot. **Compared to the overall payload of the robot this weight allows enough working space (approx.** $700mm$ **in x and y directions) combined with high dynamics (approx. maximum flange velocity of** $35m/s$**, maximum flange acceleration of** $40m/s^2$**).** Three different cases have been investigated. First, there was no additional weight during the learning phase, but it was added afterwards before the loop was closed and the evaluation phase took place. Figure 7a shows the results. The controller was not able to cope with it. The trajectory spiraled slowly away to the center of the working space. In the second case, the weight was added during learning, but was removed for the evaluation (see Figure 7b). The controller had no significant problem to cope with this unexpected change. Finally, in a third experiment, the weight was in place for both phases, i.e., during learning and evaluation, and not surprisingly, the system was able to learn the right output weights to successfully control the robot arm.

A further set of experiments was carried using additional target trajectories defined in Table 1. The results are summarized in Figure 8. The left column shows the performances in the learning phase (open-loop) and the right column during the evaluation phase (closed-loop).

For the oscillatory axial movement, Figure 8a and 8b, the approach is able to learn to represent the target trajectory and the 2D plots indicate a correct movement. However, during evaluation (closed-loop) the actual muscle pressure values reveal how difficult the task seems to be for the

[1] The low update rate is due to the slow motion of the robot.

[1] **Note that the approach is very general and a mathematical proof from [10] suggests that theoretically any continuous and smooth trajectory can be implemented. The actual limitations are introduced by the mechanical structure itself, since it acts as a low pass filter. This limits the speed of the trajectory and how sharp turns can be.**

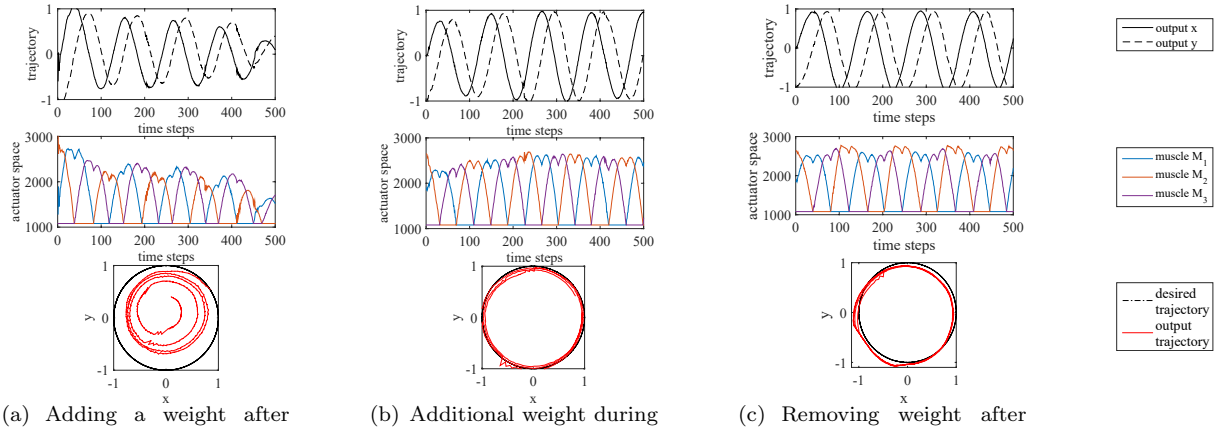


(a) Learning performance (open loop)

(b) Evaluation performance (closed loop)

(c) Testing robustness by fixing arm to the center

Figure 6. Results of experiments with circular target trajectory (see Table 1). (a) Performance in open loop, i.e., the trajectory produced offline by using collected data from matrix $\mathbf{L}_{clear}$. (b) Performance in the evaluation phase, i.e., closed loop. (c) Robustness test by fixing the arm in the evaluation phase to the center position and releasing it at $t = 250$



(a) Adding a weight after learning

(b) Additional weight during learning and evaluation

(c) Removing weight after learning

Figure 7. Results testing robustness by adding weight to the final segment of the robot arm. (a) No weight during learning, but added in evaluation phase (b) **weight during learning and evaluation** (c) **weight during learning, but removed for evaluation phase**

control setup. It oscillates back and forward exhibiting very inefficient behaviour. It seems the system is not able to detect when to change direction.

Since the circular movement was learned successfully, we investigated if it is possible to approximated the axial movement by deforming the circle into ovals. We reduced stepwise the ratio between the minor and the major axis of the oval until the robot showed the same unpredictable behaviour as for the axial movement. The closest approximation that could be still successfully learned was the oval trajectory depicted in Figures 8c and 8d. In this case the minor axis was 10% of the major axis.

The last experiments (Figures 8e and 8f) was to learn a logarithmic spiral movement. Note that in contrast to all previous tasks this trajectory does not repeat oneself, i.e., it is not a limit-cycle. Although the system is far off from the target trajectory, remarkably, it follows a scaled version of it with an offset.

5. Conclusion and Future work

We have implemented a morphological computation control approach as suggested by Hauser et al. [10] in a highly complex, pneumatically driven, modular robot arm. The controller was model-free and the feedback was learned in a supervised learning setup and with simple linear regression. We demonstrated that circular (oval) movements can be successfully implemented and the learned controller exhibits robustness to a certain extent. **Note that the system is able to learn a robust trajectory, i.e., learn a region of attraction around the desired trajectories, only because of the reservoir, i.e., the rich dynamics of the arm itself that are exploited. As pointed out in [12] this would not be possible with a simple feedforward control.**

We also identified a number of limitations of the straight forward implementation. For example, the task to follow straight lines seems to be hard to achieve in this particular robot arm. This might be due to the fact that we have three pneumatic muscles, which, although symmetrically arranged, do not have the exact same dynamic properties. Results on oval movements suggest that a simple solution could be to use a slightly bended trajectory, which seems to be more 'natural' to the physicality of the robot arm.

Another limitation was to follow a spiral movement. The evaluation of the learned controller showed a trajectory that is far off the target trajectory, but seems to be a scaled version of the spiral trajectory in addition to an offset. Again, it seems the underlying assumption of symmetry of the physical robot arm is not correct.

Another limitation seems to be coming directly from the proposed mapping from Cartesian coordinates to the pneumatic muscle pressures as defined in Figure 5. This assumes a smooth and perfect mapping is possible and is indeed achieved by the derived equations. Tracking the actual movement with an electromagnetic sensor[1] attached to the last segment revealed that this assumption is not true. Figure 9 shows the actual trajectory for the circle task. Clearly, the trajectory is distorted and the equations from Figure 5 do not reflect the actual function.

However, both issues, the scaling of the spiral as well as the mis-mapping to the actual operational space could be potentially solved by extending the morphological computation control framework. Since the learning task only consists of finding sets of linear weights, one could easily use online regression algorithms to optimize the readout to reflect this nonlinear feature of the setup. This has been previously done in simulation, e.g., [24], and will be part of future work. **Online regression algorithms would also provide a framework to learn to improve on a given control signal. This is particular interesting, when it's hard to find a good supervised learning data set.**

Considering the safety concern discussed in the introduction, there seems to be a trade-off between precise control and softness. Consequently, a solution would be to have soft robotic arms that can change their stiffness depending on the working situation. For example, for longer trajectories, e.g., to move an object from A to B, a compliant structure that is not precisely controlled can be more than sufficient while being safer to interact. On the other for a precision task, like fitting a piece, a rigid arm controlled with a conventional controller would be a good choice. It seems a combination of traditional control approach, morphological computation based approach and the capability to adapt the level of rigidness could potentially provide the solution for safe, close human-robot collaboration.

[1]The sensor was a coil that was detected by means of a stationary fixed electromagnetic transmitter box below the robot arm.

6. Acknowledgments

This work was supported by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 608022 (SMART-E) and grant agreement no. 619319 (RoboSoft).

References

[1] Fumiya Iida and Cecilia Laschi. Soft robotics: Challenges and perspectives. *Procedia Computer Science*, 7:99–102, 2011.

[2] Haider Abidi and Matteo Cianchetti. On intrinsic safety of soft robots. *Frontiers in Robotics and AI*, 4:5, 2017. doi: 10.3389/frobt.2017.00005.

[3] ISO. *ISO 10218-1 (2012): Robots and robotic devices - Safety requirements for industrial robots - Part 1: Robots*. International Organization for Standardization, Geneva, Switzerland, January 2012.

[4] ISO. *ISO/TS 15066 (2016): Robots and robotic devices - collaborative robots*. International Organization for Standardization, Geneva, Switzerland, February 2016.

[5] ISO. *ISO 10218-2 (2012): Robots and robotic devices - Safety requirements for industrial robots - Part 2: Robot systems and integration*. International Organization for Standardization, Geneva, Switzerland, June 2012.

[6] ISO. *ISO 13849-1 (2016): Safety of machinery - Safety-related parts of control systems - Part 1: General principles for design*. International Organization for Standardization, Geneva, Switzerland, June 2016.

[7] Jeffrey Too Chuan Tan, Feng Duan, Ye Zhang, Ryu Kato, and Tamio Arai. Safety design and development of human-robot collaboration in cellular manufacturing. In *Automation Science and Engineering, 2009. CASE 2009. IEEE International Conference on*, pages 537–542, 2009.

[8] Rolf Pfeifer and Josh C. Bongard. *How the Body Shapes the Way we Think*. The MIT Press, 2006. ISBN 0262162393.

[9] Helmut Hauser, Auke J. Ijspeert, Rudolf M. Füchslin, Rolf Pfeifer, and Wolfgang Maass. Towards a theoretical foundation for morphological computation with compliant bodies. *Biological Cybernetics*, 105(5):355–370, 2011. doi: 10.1007/s00422-012-0471-0.

[10] Helmut Hauser, Auke J. Ijspeert, Rudolf M. Füchslin, Rolf Pfeifer, and Wolfgang Maass. The role of feedback in morphological computation with compliant bodies. *Biological Cybernetics*, 106(10): 595–613, 2012.

[11] Keyan Zahedi and Nihat Ay. Quantifying morphological computation. *Entropy*, 15(5):1887–1915, 2013.

[12] Helmut Hauser, Kohei Nakajima, and Rudolf M Füchslin. Morphological Computation – The Body as a Computational Resource. In Helmut Hauser, Rudolf M Füchslin, and Rolf Pfeifer, editors, *E-book on Opinions and Outlooks on Morphological Computation*, chapter 20, pages 226–244. 2014. ISBN 978-3-033-04515-6.

[13] Steve Collins, Andy Ruina, Russ Tedrake, and Martijn Wisse. Efficient Bipedal Robots Based on Passive-Dynamic Walkers. *Science*, 307:1082–1085, 2005.

[14] Kohei Nakajima, Helmut Hauser, Rongjie Kang, Emanuele Guglielmino, Darwin G Caldwell, and Rolf Pfeifer. A Soft Body as a Reservoir: Case Studies in a Dynamic Model of Octopus-Inspired Soft Robotic Arm. *in Frontiers Computational Neuroscience, Research Topic: Modularity in motor control: from muscle synergies to cognitive action representation*, 7(91):91, jun 2013. ISSN 16625188. doi: 10.3389/fncom.2013.00091.

[15] Kohei Nakajima, Tao Li, Helmut Hauser, and Rolf Pfeifer. Exploiting short-term memory in soft body dynamics as a computational resource. *Journal of The Royal Society Interface*, 11(100):20140437, 2014. doi: 10.1098/rsif.2014.0437.

[16] Kohei Nakajima, Helmut Hauser, Tao Li, and Rolf Pfeifer. Information processing via physical soft body. *Scientific reports*, 5:10487, 2015. ISSN 2045-2322. doi: 10.1038/srep10487.

[17] Ken Caluwaerts, Jérémie Despraz, Atıl Işçen, Andrew P Sabelhaus, Jonathan Bruce, Benjamin Schrauwen, and Vytas SunSpiral. Design and control of compliant tensegrity robots through simulation and hardware validation. *Journal of the Royal Society Interface*, 11(98):20140520, 2014.

[18] Qian Zhao, Kohei Nakajima, Hidenobu Sumioka, Helmut Hauser, Rolf Pfeifer. Spine Dynamics As

a Computational Resource in Spine-Driven Quadruped Locomotion. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[19] Benjamin Schrauwen, David Verstraeten, and Jan Van Campenhout. An overview of reservoir computing: theory, applications and implementations. In *Proceedings of the 15th European Symposium on Artificial Neural Networks. p. 471-482 2007*, pages 471–482, 2007.

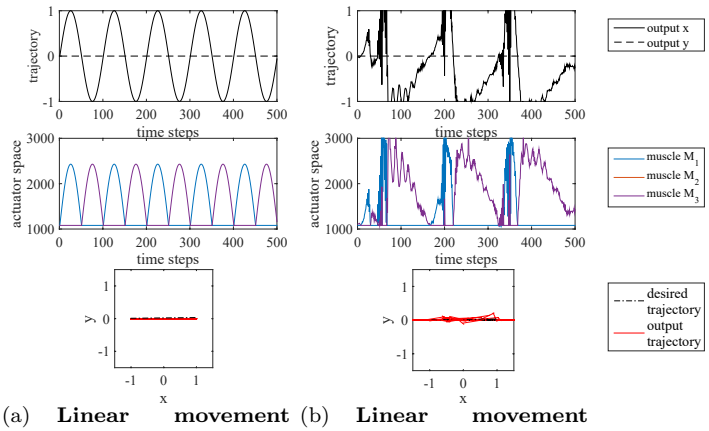
[20] Martin Eder, Maximilian Karl, Alois Knoll, and Stefan Riesner. Compliant worm-like robotic mechanism with decentrally controlled pneumatic artificial muscles. In *Innovative Engineering Systems (ICIES), 2012 First International Conference on*, pages 243–248, 2012.

[21] Martin Eder, Maximilian Karl, Felix Schultheiß, Johannes Schurmann, Alois Knoll, and Stefan Riesner. Design of an inherently safe worm-like robot. In *Safety, Security, and Rescue Robotics (SSRR), 2013 IEEE International Symposium on*, pages 1–6, 2013.

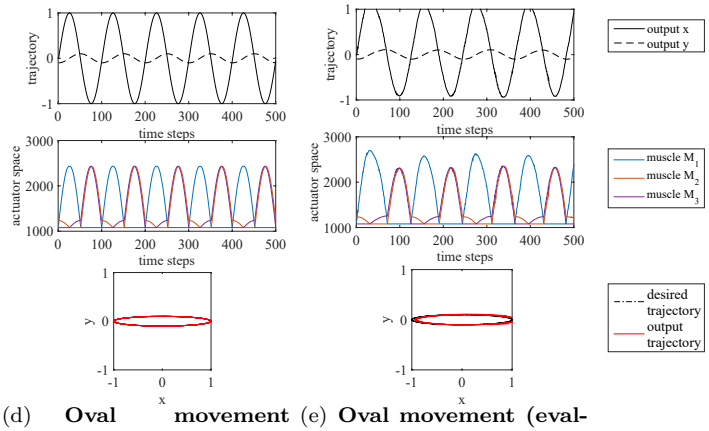
[22] Martin Eder, Maximilian Karl, Alois Knoll, and Stefan Riesner. Continuum worm-like robotic mechanism with decentral control architecture. In *Automation Science and Engineering (CASE), 2014 IEEE International Conference on*, pages 866–871, 2014.

[23] Dieter Bergemann, Bernd Lorenz, and Axel Thallemer. Actuating means, 2002. US Patent 6,349,746.

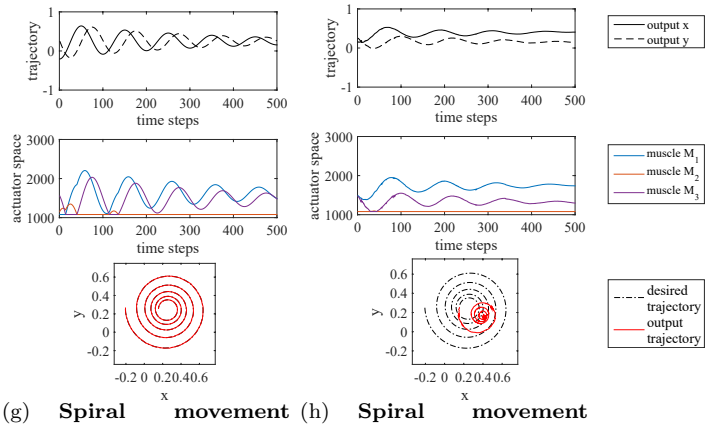
[24] Ken Caluwaerts, Michiel D'Haene, David Verstraeten, and Benjamin Schrauwen. Locomotion without a brain: physical reservoir computing in tensegrity structures. *Artificial life*, 19:35–66, 2013. ISSN 10645462.



(a) **Linear movement (learning)** (b) **Linear movement (evaluation)**



(d) **Oval movement (learning)** (e) **Oval movement (evaluation)**



(g) **Spiral movement (learning)** (h) **Spiral movement (evaluation)**

Figure 8. Results on various trajectories **(linear, oval and spiral movements)**. Left column is the learning performance, right column is evaluation performance.**The evaluation (b) of the linear movement shows that the controller is not able to handle overlapping trajectories. Introducing an additional lateral movement (oval trajectory (c,d)) solves this issue. The controller can also handle trajectories which are not limit-cycles up to a point (e,f).**

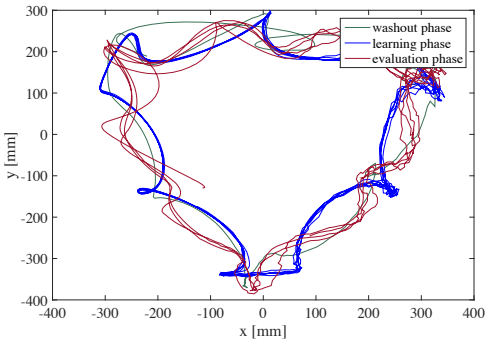


Figure 9. Sensor data from electromagnetic tracking device from the movement on a circle trajectory.